dunetpc - Task #23177

Modify protoDUNE SP dataprep to use the decoder tool instead of reading digits from the event store

08/27/2019 04:46 PM - David Adams

Status:	Closed	**Start date:**	08/27/2019
Priority:	Normal	**Due date:**	
Assignee:	Tingjun Yang	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Tom has created a tool that decodes protoDUNE raw data and returns it to the caller in the form of vector<raw::RawDigit>. Tingjun has requested that protoDUNE dataprep be modified to use this tool instead of reading digits from the event data store.

I am working on this.

History

#1 - 08/28/2019 12:14 PM - David Adams

I have committed an update of DataPrepModule that provides the option to take raw digits and their status (RDStatus) from the tool instead of the data store. The configuration prolog producer_adcprep is replaced with producer_adcprep_notool that uses the data store (as before) and producer_adcprep_tool that uses the tool. I have verified that I can drop the TPC decoder producer if the latter configuration is used. The old name points back to the first configuration but we can change this when we decide we prefer the second.

#2 - 09/06/2019 04:05 PM - David Adams

This seems to work fetching all data before looping over APAs but the memory usage is the same.

Tom indicates we should fetch one APA at a time.

#3 - 10/21/2019 08:15 AM - David Adams

I have created a new module DataPrepByApaModule that processes one APA at a time. It also drops some of the extra baggage added to DataPrepModule. I does allow users to specify which channel groups or ranges to process. Use ChannelGroups: ["all"] for the full detector all at once or ChannelGroups: ["apas"] for the full detector, one APA at a time.

With that option and running dataprep with no tools, I see VM/RSS = 1.32/0.68 GB for the old module or the first option and 1.19/0.55 with the second option, i.e. a saving of 130 MB. Same results for 1 and 10 events.

I also looked at the time spent in the dataprep module to process a single APA again with no tools. The old module (reading in all APAs) requires an average of 6.9 sec/evt and the new require 1.4 sec/event. The last number drops to 1.2 sec/event the second time I process the same batch of events. Presumably this is because that part of the file is cached in memory.

The new module is not yet pushed.

#4 - 10/21/2019 09:58 AM - David Adams

Correction: the new module is pushed.

It may not yet be correctly writing output containers. I am working on that.

I propose to write digits and time stamps for all channels in APAs that are decoded and to write wires for all channels that have those after processing. The names for the three containers are configurable and, if any is blank, the corresponding container is not written. Presumably, production will want to write time stamps and wires.

#5 - 10/21/2019 11:31 AM - Thomas Junk

Very good!

#6 - 10/21/2019 01:29 PM - David Adams

I have pushed some changes that should fix the writing time stamps, digits and wires, all controlled separately with container names. I verified that when I process all but five channels, I get 15360 time stamps, 15360 digits and 15355 wires.

Tingjun or Christoph, I would like to run the CI tests with the new module. I think we need to replace producer_adcprep with producer_adcprep_byapa or I can change the definition of producer_apa in dataprep_dune.fcl. Should I do the latter? It is easy to switch back.

#7 - 10/21/2019 02:38 PM - Tingjun Yang

David Adams wrote:

> I have pushed some changes that should fix the writing time stamps, digits and wires, all controlled separately with container names. I verified that when I process all but five channels, I get 15360 time stamps, 15360 digits and 15355 wires.
>
> Tingjun or Christoph, I would like to run the CI tests with the new module. I think we need to replace producer_adcprep with producer_adcprep_byapa or I can change the definition of producer_apa in dataprep_dune.fcl. Should I do the latter? It is easy to switch back.

Hi David,

I would suggest to make the change to
dunetpc/fcl/protodune/reco/protoDUNE_reco_data_Dec2018.fcl (line 41)

This will only change the configuration for data reco and leave MC reco unchanged (assuming this is what we want for now).

Thanks,
Tingjun

#8 - 10/22/2019 09:01 AM - David Adams

Good point about the MC. I see the file you reference is prolog only. I was confused for a while try to use it as a top-level fcl. I suggest to rename it protoDUNE_reco_data_prolog.fcl.

I made the change you suggest and ran with protoDUNE_reco_data.fcl and get this error:

```
CookedFrameSource failed to get raw::RawDigits: caldata
```

followed by an exception.

The new module can but is not configured to write the raw digits because I understood subsequent reco only needs the wires. Does the wirecell code really need the raw digits?

da

#9 - 10/22/2019 09:12 AM - Tingjun Yang

Yes, GausHitFinder used to require raw digits but it has been changed #23196.

I am not sure if wirecell requires raw digits. I have added Wenqiang as a watcher.

I will try to make some changes to the fcl parameters to see if it works without rawdigits.

#10 - 10/22/2019 09:27 AM - Wenqiang Gu

Hi, *wirecell* does not require raw digits in the current configuration. The prompt message might be a little bit confusing, but the "CookedFrameSource" means the cooked raw data from "dataprep". It is looking for a data product of recob::Wire with label "caldata". You can find the detailed configuration in wirecell_dune.fcl around line 85.

#11 - 10/22/2019 09:36 AM - Tingjun Yang

Thanks Wenqiang.

David, you need to change line 83 of dune/DUNEWireCell/wirecell_dune.fcl to the label of recob::Wire produced by dataprep. I guess it is different from caldata now (maybe with an instance name).

#12 - 10/22/2019 09:41 AM - David Adams

If the code is looking for recob::Wire, then the error message should be fixed to say this. I suspected as much and tried changing the wire label to caldata but get the same error.

Art identifies data products with the producer name and an instance name which can be blank. Are you saying I should use producer:instance = "caldata:", i.e. the producer name "caldata" and instance name ""?

I have tried "caldata:dataprep" and "caldata:caldata" without success.

If so, do you support a non-blank instance name? If not, I will have to modify my code to produce such. Right now, I use blank to mean no output.

Thanks.

da

#13 - 10/22/2019 09:43 AM - David Adams

Tingjun's comment came it as I was writing. I will try his suggestion. Thanks. --da

#14 - 10/22/2019 09:51 AM - Wenqiang Gu

Hi David,

I will fix the message in the later version.

Please take a look at this line:
https://cdcvs.fnal.gov/redmine/projects/larwirecell/repository/revisions/master/entry/larwirecell/Components/CookedFrameSource.cxx#L101

The "m_inputTag" is the product name that you assigned in wirecell_dune.fcl

--Wenqiang

#15 - 10/22/2019 10:03 AM - David Adams

I added this to my configuration:

```
physics.producers.wclsdatasp.wcls_main.params.raw_input_label: "caldata:dataprep"
```

and instead of the above error, get an indication of success:

```
CookedFrameSource: got 15360 raw::RawDigit objects
    input nticks=6000 keeping as is
Retagger: tagging trace set: wiener with 108321 traces, 0 summary
Retagger: tagging trace set: gauss with 84038 traces, 0 summary
wclsFrameSaver saving cooked to 6000 ticks
wclsFrameSaver: saving 84038 traces tagged "gauss"
FrameSaver: q=2.66116e+09 n=1495078 tag=gauss
wclsFrameSaver: saving 108321 traces tagged "wiener"
FrameSaver: q=2.85849e+09 n=1349881 tag=wiener
```

Presumably the message should read recob::Wire instead of raw::RawDigit.

However, after this, I get an error complaining of missing raw::RawDigit. I am chasing that down.

Wenqiang, could you make sure that the messages that make it to our log file, clearly indicate their source. Ideally precede them with full class name (including namespace), e.g.

```
wirecell::FrameSaver:: q=2.66116e+09 n=1495078 tag=gauss
```

Thanks. --da

#16 - 10/22/2019 10:09 AM - Wenqiang Gu

Hi David,

I assume wirecell signal processing is successful now if you see these messages.

```
wclsFrameSaver: saving 84038 traces tagged "gauss"
FrameSaver: q=2.66116e+09 n=1495078 tag=gauss
wclsFrameSaver: saving 108321 traces tagged "wiener"
FrameSaver: q=2.85849e+09 n=1349881 tag=wiener
```

Sure, I will change the warning message to be more friendly as you suggested.

#17 - 10/22/2019 10:09 AM - Tingjun Yang

I think I know how to fix the rawdigit issue. I will post a solution soon.

#18 - 10/22/2019 10:23 AM - David Adams

OK. FWIW, I do not get a crash if I use this producer list:

```
physics.reco: ["caldata", "wclsdatasp"]
```

and do get a crash with this list:

```
physics.reco: ["caldata", "wclsdatasp", "digitwire"]
```

If you think we are ready, please put in all the mods and we can see how the CI does. If we don't need to write raw digits, I am ready to go.

#19 - 10/22/2019 10:30 AM - Tingjun Yang

Hi David,

Yes, the problem is related to "digitwire", which is just a simply module to wrap recob::Wire and raw::RawDigit as inputs to GausHitFinder.

I tried to exclude digitwire and use wclsdatasp as input to GausHitFinder, but it seems that GausHitFinder still requires the presence of raw::RawDigit. #23196 is not really resolved. I have reopened that issue and I am afraid we need to wait for that issue to be resolved first before we can enable tool-based dataprep.

#20 - 10/22/2019 11:20 AM - David Adams

- Assignee changed from David Adams to Tingjun Yang

Understood. The new dataprep module can write digits if they are needed but it looks like that problem is now with the associations which should not be needed as both RawDigit and Wire hold their channel numbers.

I have pushed all my changes without the changes to high-level fcl. Those can be inferred from the fcl I use to demonstrate running with the new dataprep module:

```
#include "protoDUNE_reco_data.fcl"
```

```
physics.producers.caldata: @local::producer_adcprep_byapa
```

```
physics.producers.wclsdatasp.wcls_main.params.raw_input_label: "caldata:dataprep"
```

I add the following to make things quicker and avoid crashes:

```
physics.reco: ["caldata", "wclsdatasp", "digitwire", "gaushit"]
physics.reco: ["caldata", "wclsdatasp", "digitwire"]
physics.reco: ["caldata", "wclsdatasp"]
```

Once you have the digitwire fix, please try the above or its equivalent in reco fcl. When all is working, please commit the changes and report here. Let me know if I should help.

Thanks.

#21 - 10/22/2019 11:23 AM - Tingjun Yang

Hi David,

I will change the reco fcl parameters as you instructed once the gaushit rawdigit issue is resolved.

Thanks.

#22 - 11/05/2019 01:55 PM - Tingjun Yang

#23196 is resolved and will be integrated into this week's larsoft release.

I have prepared feature branch feature/tjyang_removerawdigits. This feature branch removed EventButcher from reco chain and use the output from WireCellToolkit after scaling it. The reconstruction and event display fcl files have been modified accordingly.

With the current develop head, the memory usage of reconstructing run: 5809 subRun: 1 event: 4503 is:

```
  Peak virtual memory usage (VmPeak)  : 2985.65 MB
  Peak resident set size usage (VmHWM): 1956.02 MB
```

With the feature branch, the memory usage is:

```
  Peak virtual memory usage (VmPeak)  : 2893.87 MB
  Peak resident set size usage (VmHWM): 1880.89 MB
```

Excluding EventButcher saves some memory which is expected. The first step is complete. I am going to working on integrating tool-based raw decoder, which does not create raw digits.

#23 - 11/05/2019 04:33 PM - Tingjun Yang

Feature branch feature/tjyang_removerawdigits is updated to use caldata: @local::producer_adcprep_byapa and remove tpcrawdecoder from reco chain.

The memory usage is further reduced for the same event:

```
  Peak virtual memory usage (VmPeak)  : 2709.99 MB
  Peak resident set size usage (VmHWM): 1693.07 MB
```

Two things to note:
1. Now raw::RDTimeStamp for each digit has the label: "caldata:dataprep".

2. Before the change, tpcrawdecoder + caldata takes 11 + 30 = 41 s. After the change, caldata takes 67 s. It is a little longer, but the gain in memory reduction is significant.

#24 - 11/09/2019 07:26 AM - Tingjun Yang

- % Done changed from 0 to 100

- Status changed from Work in progress to Resolved

Available in dunetpc v08_35_00.

#25 - 11/13/2019 10:21 AM - David Adams

The change in reco time is surprising. I tried running no tools and the standard dataprep tools (protodune_dataprep_tools_wirecell) with the old and new dataprep module. The old/new times were 6.3/6.5 sec for no tolls and 40.7/40.1 sec for full dataprep. By eye, most of the reco time is spent in the pedestal finder.

#26 - 11/14/2019 01:13 PM - David Adams

- Status changed from Resolved to Feedback

In the results reported in the preceding message, I was not writing out any event data or even creating the output wires. I enabled the wire building and event data output and the reported time for the dataprep module only increased from 41.5 to 43.4 sec/event. I do not save data with the old module so the latter number cannot be compared. But it is clear that the digit building and insertion into the event only adds one or two seconds and so I continue testing without the output.

My "old" results above used the old dataprep module with the decoder tool (not module). If I go back and run with the decoder module, the dataprep time falls to 37.0 sec/event but the module takes 6.5 sec/event, a total of 43.5 sec/event. The new dataprep (including decoding) uses 41.5 sec/event. These tests were run with the first five events in np04_raw_run005152_0001_dl1.root.

Tingjun, I cannot reproduce the slowdown you report. Can you see if you can repeat it and send me your config so I can try as well? It would be very helpful if you can disable all downstream processing.

Thanks.

da

#27 - 07/03/2020 10:07 AM - David Adams

- Status changed from Feedback to Closed

I suppose this is resolved. Reopen if not.